Persönliche Codes bei Längsschnittuntersuchungen III: fehlertolerante Zuordnung unverschlüsselter und verschlüsselter selbstgenerierter Codes im empirischen Test

Pöge, Andreas

Veröffentlichungsversion / Published Version
Zeitschriftenartikel / journal article

Zur Verfügung gestellt in Kooperation mit / provided in cooperation with:
GESIS - Leibniz-Institut für Sozialwissenschaften

Empfohlene Zitierung / Suggested Citation:
Pöge, A. (2011). Persönliche Codes bei Längsschnittuntersuchungen III: fehlertolerante Zuordnung unverschlüsselter und verschlüsselter selbstgenerierter Codes im empirischen Test. *Methoden, Daten, Analysen (mda)*, *5*(1), 109-134. https://nbn-resolving.org/urn:nbn:de:0168-ssoar-255044

Persönliche Codes bei Längsschnitt-untersuchungen III

Personal Codes in Longitudinal Studies III

Fehlertolerante Zuordnung unverschlüsselter und verschlüsselter selbst-generierter Codes im empirischen Test

The Empirical Test of Fault-Tolerant Linkage of Unencrypted and Encrypted Self-Generated Codes

Andreas Pöge

Zusammenfassung

In Längsschnittuntersuchungen werden oftmals selbstgenerierte persönliche Codes verwendet, um Daten aus mehreren Erhebungszeitpunkten miteinander zu verknüpfen. Aus Gründen der Ausschöpfung und Datenqualität muss die Zuordnungsmethode dabei fehlertolerant sein. In diesem Artikel wird die Qualität der fehlertoleranten Zuordnung von *verschlüsselten* Codes mit der von *unverschlüsselten* Codes verglichen. Um geeignete Daten zu erheben, wurde ein Feldexperiment mit Studierenden durchgeführt. Für die Zuordnung der verschlüsselten Codes wurden die von Schnell, Bachteler und Reiher (2009a) entwickelten und in dieser Zeitschrift vorgestellten Programme „Merge Toolbox", „BloomEncoder" und „Bloom-Comparator" eingesetzt. Die Ergebnisse der Analysen zeigen, dass die fehlertolerante Zuordnungsmethode mit verschlüsselten Codes herkömmlichen Methoden mit unverschlüsselten sogar überlegen ist.

Abstract

In this article the quality of a fault-tolerant linkage of unencrypted and encrypted self-generated codes is compared. To obtain suitable data a field experiment with students was carried out. For the linkage of the encrypted codes the programs "Merge Toolbox", "BloomEncoder" and "BloomComparator" developed by Schnell, Bachteler and Reiher (2009a) were used. The results show that the fault-tolerant linkage with encrypted codes works even better than using unencrypted codes under special conditions.

1 Einleitung

In Panelerhebungen mit sensiblem Befragungsthema besteht oftmals die Notwendigkeit, die Befragten, die zu mehreren Zeitpunkten teilnahmen, einander mit Hilfe von selbstgenerierten persönlichen Codes zuzuordnen. Zum Teil ergibt sich dies aus Gründen der Ausschöpfungsoptimierung, indem durch Zusicherung der Anonymität positive Effekte auf die Teilnahmebereitschaft erzielt werden sollen, zum Teil auch aus datenschutzrechtlichen Vorgaben. Im Bereich der Sozialwissenschaften werden zu diesem Zweck häufig Codes eingesetzt, die durch Fragen zu zeitstabilen persönlichen Merkmalen gebildet werden (vgl. Grube/Morgan/Kearney 1989: 159). Die Daten aus unterschiedlichen Erhebungszeitpunkten werden dann über die gebildeten Codes zugeordnet. Dieses Verfahren ist aus unterschiedlichen Gründen nicht unproblematisch: Der Code muss über eine ausreichende Anzahl geeigneter Fragen gebildet werden, so dass mit ausreichender Sicherheit gewährleistet ist, dass unterschiedliche Befragte auch unterschiedliche Codes aufweisen. Entscheidend ist hier das Zusammenspiel aus der Länge des Codes und der Varianz der einzelnen Stellen bzw. der möglichen Antworten zu den diesbezüglichen Fragen (vgl. Pöge 2005b, 2008).

Darüber hinaus zeigt die Erfahrung, dass die konsistente Beantwortung der persönlichen Fragen zu mehreren Zeitpunkten häufig in nicht unbeträchtlichem Ausmaß scheitert. Dies führt zu „fehlerhaften" Codes, wodurch eine fehlertolerante Zuordnung nötig wird, soll nicht auf einen erheblichen Teil der Befragungspersonen – mit möglicherweise negativen Auswirkungen auf die Datenqualität[1] – verzichtet werden (vgl. detailliert Pöge 2005b: 66f., 2008: 67; Galanti/Siliquini/Cuomo u. a. 2007; Yurek/Vasey/Havens 2008). Im Zusammenhang mit der geforderten Codelänge ist hierbei zu bedenken, dass auch die fehlerhaften Codes eindeutig sein müssen, so dass nicht mehrere Personen dieselben Codes aufweisen. Konkret heißt das, mehrere Codes sollten nicht durch zufällige Fehler an ein oder mehreren Stellen identisch werden, obwohl sie unterschiedlich sein müssten. Wendet man eine einfache fehlertolerante Zuordnung an, bei der Fehler zugelassen werden, indem einzelne Codestellen bei der Zuordnung ignoriert werden, müssen die Codes auch bei Reduktion um die fehlerhafte Stelle eindeutig bleiben.

Reicht diese Form der Anonymisierung über selbstgenerierte persönliche Codes nicht aus, wenn etwa bei besonders sensiblem Untersuchungsgegenstand

1 Es sind unter anderem Verzerrungen im Hinblick auf Geschlecht und (Schul-) Bildung zu erwarten. Dies kann sich beispielsweise verzerrend auf Kriminalitätsraten auswirken (siehe Pöge 2005b: 66f., 2008: 67).

(Kriminalität, Sexualität, Gesundheit etc.) und/oder besonders schützenswerter Population (zum Beispiel minderjährige Befragte) befürchtet wird, aus dem gebildeten Codewort ließen sich Rückschlüsse auf persönliche Merkmale ziehen und somit eine Identifizierung der Probanden ermöglichen, kann eine zusätzliche Verschlüsselung der Codes gewünscht oder sogar gefordert sein.[2] Gleiches gilt für Zuordnungsverfahren, welche auf „Codes" basieren, die aus persönlichen Merkmalen wie beispielweise Namen, Adressen, Geburtsorten etc. in Reinform bestehen und die trotzdem über eine Verschlüsselung ein sehr hohes Maß an Anonymität sichern möchten (siehe Schnell/Bachteler/Reiher 2009a: 204). Auch mit diesen Merkmalen kann es bei mehrfacher Erhebung zu fehlerbehafteten Codes kommen. Sollen diese Codes verschlüsselten einander zugeordnet werden, ist auch hier ein fehlertolerantes Verfahren notwendig.

Bislang war das Wissen über Zuordnungsmethoden mit verschlüsselten Codes limitiert. Von Schnell, Bachteler und Reiher (2009a, b) wurde nun jedoch unlängst ein Verfahren vorgestellt, welches solch ein Vorgehen ermöglicht. Die Autoren stellen darüber hinaus im Rahmen ihres SAFELINK-Projektes mit den frei verfügbaren, plattformübergreifenden Java-Programmen „BloomEncoder" und „BloomComparator"[3] Software zur Verfügung, mit der sowohl die Verschlüsselung als auch die Zuordnung von Codes für die Anwenderin bzw. den Anwender komfortabel handhabbar wird.

Dieses im nächsten Abschnitt näher erläuterte Verfahren soll hier getestet werden. Dabei wird so vorgegangen, dass Codes, die mit Hilfe eines Feldexperimentes gewonnen wurden, zum einen unverschlüsselt und zum anderen über das SAFELINK-Verfahren verschlüsselt zugeordnet werden. Beide Herangehensweisen sollen im Hinblick auf die Zuordnungsperformanz verglichen werden. Es wird herausgearbeitet, dass der Einsatz von verschlüsselten Codes mit dem SAFELINK-Verfahren praktikabel ist und deutliche Vorteile für die Zuordnung im Vergleich zu einem Vorgehen auf Basis von unverschlüsselten Codes bietet. Da der in dem genannten Feldexperiment eingesetzte Code in ähnlicher Weise seit dem Jahr 2002 in dem DFG-Projekt „Kriminalität in der modernen Stadt" eingesetzt wird (siehe Pöge 2005b, 2008), können aufschlussreiche Vergleiche mit der Praxis angestellt werden. Des Weiteren sollen in dieser Arbeit Anwendungsempfehlungen für den Einsatz des SAFELINK-Verfahrens und die bei der eingesetzten Software einzustellenden Parameter entwickelt werden.

2 Diese Variante war in dem DFG-geförderten Forschungsprojekt „Kriminalität in der modernen Stadt" diskutiert worden (vgl. Pöge 2005b, 2008), ließ sich jedoch wegen der damals noch nicht zur Verfügung stehenden Möglichkeiten nicht realisieren.

3 Die genannten Programme können über http://www.uni-due.de/soziologie/schnell_forschung_safelink.php bezogen werden.

1.1 Das SAFELINK-Verfahren

Das SAFELINK-Verfahren basiert darauf, dass Codewörter zunächst in sogenannte *N-Gramme*, das heißt eine Folge aus *N* Zeichen, zerlegt werden. Die Anzahl der Zeichen kann dabei frei bestimmt werden. Daneben kann ausgewählt werden, ob dem ersten und letzten Teilstring ein Leerzeichen voran- bzw. hintangestellt wird. Setzt man die Zahl der Zeichen beispielsweise auf 2, verwendet man also sogenannte *Bigramme*, und verwendet die Leerzeichen zu Beginn und Ende, so würde ein hypothetisches Codewort „ABC1DEF" zerlegt in: „_A AB BC C1 1D DE EF F_". Diese Teilstrings werden dann mit einer wählbaren Anzahl an *kryptografischen Hashfunktionen*[4] jeweils in Form von Einsen an bestimmten Positionen in einem sogenannten *Bloomfilter* mit ebenfalls wählbarer Länge gespeichert (vgl. Schnell/Bachteler/Reiher 2009a: 207ff.). Im Ergebnis wird somit jedes Codewort in einen Bitvektor, also einen Vektor, bestehend nur aus Nullen und Einsen, mit vorzugebender Länge umgewandelt und kann bei einer genügend großen Zahl an eingesetzten Hashfunktionen nicht mehr rekonstruiert werden. Dadurch, dass die Codewörter zunächst in N-Gramme aufgespalten und dann erst im Bitvektor gespeichert werden, ergibt sich trotzdem die Möglichkeit der fehlertoleranten Zuordnung. Wenn nämlich beispielsweise nur an einer Stelle ein „Fehler" zwischen zwei zuzuordnenden Codes besteht, werden nur die N-Gramme, die diese fehlerhafte Codestelle enthalten, an unterschiedlichen Positionen auf dem Bitvektor gespeichert. Die fehlerfreien Stellen werden nach wie vor an denselben Positionen in den Bitvektor als Einsen geschrieben. Ähnliche Codes ähneln sich daher auch in der Form ihrer zugehörigen Bitvektoren (vgl. Schnell/Bachteler/Reiher 2009a: 208).

In Bezug auf die Sicherheit der Verschlüsselung ist zu bemerken, dass eine Entschlüsselung, beispielsweise durch einen illegalen sogenannten „Wörterbuchangriff"[5], immer schwieriger wird, je mehr Hashfunktionen verwendet werden (vgl. Schnell/Bachterler/Reiher 2009a: 211). Insofern ist eine hohe Zahl an Hashfunktionen aus Sicherheitsaspekten wünschenswert. Allerdings steigt bei zunehmender Zahl an Hashfunktionen auch die Wahrscheinlichkeit falsch positiver Treffer, das heißt die Identifikation von zusammengehörigen Codepärchen, die in Wirklichkeit nicht zusammengehören. Insofern muss hier eine Balance gefunden werden. Schnell, Bachteler und Reiher (2009a: 211) halten eine Hashfunktionszahl zwischen

4 Bei diesen Einwegfunktionen kann vom Ergebnis nicht mehr auf den Ausgangswert geschlossen werden (vgl. Schnell/Bachteler/Reiher 2009a: 207).

5 Die Autoren verwenden selbst den Begriff des „Wörterbuchangriffs", der allerdings nur dann kritisch wäre, wenn die zu entschlüsselnde Zeichenkette aus einer sinnvollen Zeichenkette bestünde, was hier nicht der Fall ist. In diesem Zusammenhang wäre ein „Häufigkeitsangriff" eher problematisch, der ebenfalls immer schwieriger wird, je mehr Hashfunktionen verwendet werden.

10 und 30 für vernünftig, eine Anzahl von 15 für „akzeptabel".[6] Ein Ziel der hier durchgeführten Analysen ist die Überprüfung dieser Schwellenwerte auf realer Datenbasis.

1.2 Distanzen und Ähnlichkeiten bei unverschlüsselten und verschlüsselten Codes

Um die Fälle zweier Datensätze mit Hilfe von Codes einander zuzuordnen, müssen diese Codes verglichen und deren Ähnlichkeit analysiert werden. Sind die Codes unverschlüsselt, kann im einfachsten Fall Codestelle für -stelle auf Übereinstimmung oder Fehler überprüft und die Anzahl der Fehler aufsummiert werden. Dieses Vorgehen entspricht der Ermittlung der Hamming-Distanz (HD; Hamming 1950; Leitgöb 2010: 479f.). Dieses Vorgehen kann auch als Konzeption einer Überführung des ersten in den zweiten Code betrachtet werden: Bei der Hamming-Distanz ist hierbei nur eine Operation möglich, nämlich die Substitution einer Stelle. Für jede Substitutionsoperation, die nötig ist, um den ersten in den zweiten Code zu überführen, werden Kosten von 1 berechnet. Die Hamming-Distanz kann dann für zwei Codes *a* und *b* der Länge *T* (Anzahl der Stellen) über diese Kosten bestimmt werden:

$$\mathrm{HD}(a,b) = \sum_{t=1}^{T} c_{st} \ ,$$

mit der Substitutionskostenfunktion:

$$c_{st} = \begin{cases} 0, a_t = b_t \\ 1, a_t \neq b_t \end{cases} .$$

Die maximale Distanz zwischen zwei Codes entspricht damit der Anzahl der Stellen *T* und damit der insgesamt möglichen Fehler. Bei der minimalen Distanz von 0 sind beide Codes identisch. Vergleicht man beispielsweise die fiktiven Codes „ABC1DEF" und „CAB1EFD", ergibt sich eine Distanz von 6, da 6 Stellen des Codes nicht übereinstimmen und bei einer Überführung substituiert werden müssen. Mit Hilfe der Hamming-Distanz kann auch eine auf das Intervall [0,1] normierte Hamming-Ähnlichkeit (HÄ) bestimmt werden:

$$\mathrm{HÄ} = 1 - \frac{\mathrm{HD}}{T} \ .$$

6 Diese Zahlen wurden für Trigramme entwickelt.

Im obigen Beispiel beträgt sie 1 - 6/7 = 0,143.

Eine Erweiterung der Hamming-Distanz stellt die Levenshtein-Distanz (LD; Levenshtein 1966; Leitgöb 2010: 481ff.) dar. Sie ermöglicht zusätzlich zu der Substitutionsoperation noch zwei weitere Operationen (sogenannte Indel-Operationen), das „Einfügen" und das „Löschen", für die ebenfalls (Indel-)Kosten veranschlagt werden. Mit diesen zusätzlichen Operationen wird auch der Vergleich von Strings mit ungleicher Länge möglich. Die Levenshtein-Distanz wird für zwei Codes *a* und *b* der Längen *T* und *T** rekursiv berechnet:

$$\begin{aligned}\mathrm{LD}(a_{t,}b_{t*}) &= \mathrm{d}(a_t, b_{t*}) \\ &= \min\begin{cases} d(a_{t-1}, b_{t*}) + c_i(a_t, \varphi) \rightarrow \text{Löschen von } a_t \\ d(a_{t-1}, b_{t*-1}) + c_s(a_t, b_{t*}) \rightarrow \text{Ersetzen von } a_t \text{ durch } b_t \\ d(a_t, b_{t*-1}) + c_i(\varphi, b_{t*}) \rightarrow \text{Einfügen von } b_t \end{cases}\end{aligned}$$

mit φ als Platzhalter.

Es ist bei der Levenshtein-Distanz prinzipiell möglich, unterschiedliche Kosten für die unterschiedlichen Operationen zu vergeben. Die Software „Merge Toolbox" (Schnell/Bachteler/Reiher 2005)[7], mit der die hier aufgeführten Analysen durchgeführt werden, vergibt allerdings gleiche Kosten für alle Operationen. Wird auf diese Weise die Distanz für die Codes „ABC1DEF" und „CAB1EFD" berechnet, ergibt sich eine Distanz von 4. Der erste Code kann nämlich in den zweiten Code überführt werden, indem zunächst das „C" und das „D" gelöscht (Kosten: 2) und dann an den korrekten Stellen wieder eingefügt (Kosten: 2) werden.[8] Auch mit der Levenshtein-Distanz kann analog eine auf [0,1] normierte Levenshtein-Ähnlichkeit (LÄ) bestimmt werden:

$$\mathrm{LÄ} = 1 - \frac{\mathrm{LD}}{(T + T^*)/2}.$$

Im Beispiel ergibt sich – deutlich abweichend zu der Hamming-Ähnlichkeit – eine Levenshtein-Ähnlichkeit von 1 - 4/7 = 0,429. In den Analysen von Schnell, Bachteler und Reiher (2010) zeigt die Levenshtein-Distanz bzw. -Ähnlichkeit bei der Zuordnungsermittlung mit Hilfe von selbstgenerierten Codes sehr gute Ergebnisse.

7 Das Programm kann über http://www.uni-due.de/soziologie/schnell_forschung_safelink_mtb.php bezogen werden.

8 Zur Berechnung siehe auch http://odur.let.rug.nl/~kleiweg/lev/.

Werden die Codes, die für die Zuordnung verwendet werden sollen, auf die oben geschilderte Art und Weise mit Hashfunktionen und Bloomfiltern verschlüsselt, müssen die gebildeten Filter verglichen werden. Da sie Bitvektoren, bestehend aus Nullen und Einsen, einer bestimmter Länge sind, bietet sich als Ähnlichkeitsmaß der Dice-Koeffizient (DK; Dice 1945) an. Für zwei Bloomfilter F_1 und F_2 wird er folgendermaßen bestimmt:

$$\mathrm{DK}(F_1, F_2) = \frac{2 \cdot c}{a \cdot b} \ ,$$

mit *c* als Anzahl der übereinstimmend auf eins gesetzten Bits in beiden Filtern und *a* sowie *b* als Anzahl der insgesamt auf eins gesetzten Bits in Filter 1 bzw. Filter 2. Der Dice-Koeffizient hat ebenfalls einen Wertebereich von 0 bis 1.

1.3 Zuordnungsverfahren mit unverschlüsselten und verschlüsselten Codes

In der empirischen Praxis steht eine Zuordnung über unverschlüsselte und verschlüsselte Codes vor einer Reihe von Problemen. Zunächst ist häufig nicht bekannt, von wie vielen Personen überhaupt Daten aus *beiden* Erhebungszeitpunkten (t_1 *und* t_2) vorliegen und damit auch, wie viele echt positive Treffer günstigstenfalls gefunden werden können. Darüber hinaus ist unsicher, wie fehlerhaft diese echt positiven Treffer sind, das heißt, wie viele Fehler bei einer fehlertoleranten Zuordnung zugelassen werden müssen.[9] Daneben existiert das Problem der falsch positiven Treffer: Es kann in der Praxis in durchaus beträchtlichem Ausmaß vorkommen, dass zu einem Code ein Code aus dem jeweils anderen Erhebungszeitpunkt weniger Fehler aufweist, als der korrekt dazugehörige Code, und daher möglicherweise falsch zugeordnet würde.[10] Ein falsch positiver Treffer liegt ebenfalls vor, wenn zu einem Code, der überhaupt kein passendes Pendant aus dem zweiten Erhebungszeitpunkt hat, ein Code zugeordnet würde, der zufällig auf einem relativ niedrigem Fehlerniveau vermeintlich „passt".

Ohne eine Validierung der Codezuordnungen sind die aufgezeigten Probleme kaum zu lösen. Eine Möglichkeit der Validierung besteht darin, Handschriftenvergleiche entweder der Codeblätter – wenn sie handschriftlich ausgefüllt wurden – oder der zugehörigen Fragebögen durchzuführen (vgl. Pöge 2005b, 2008). Dies

9 Nach unseren Erfahrungen sind dies mindesten zwei bzw. drei Fehler, bezogen auf einen sechs- bzw. siebenstelligen Code (vgl. Pöge 2005b, 2008 sowie die weiteren Ausführungen in diesem Artikel).

10 Mit zunehmender Länge der Codes nimmt dieses Problem allerdings deutlich ab.

ist allerdings nur dann möglich, wenn handschriftliche Angaben vorliegen und zu einem Vergleich geeignet sind (zum Beispiel offene Fragen).[11] Bei Daten, die keine handschriftlichen Angaben enthalten, kann versucht werden, eine Zuordnungsvalidierung durch die im Fragebogen gegebenen Antworten zu erreichen. Dies verspricht nur Aussicht auf Erfolg, wenn Fragen zu (relativ) zeitstabilen Merkmalen verwendet werden (beispielsweise Geschlecht, Kinderzahl, Körpergröße, Nationalität etc.). Häufig sind diese Angaben allerdings ebenfalls fehlerbehaftet bzw. werden in nicht unerheblichem Ausmaß zu zwei Erhebungszeitpunkten inkonsistent beantwortet oder sind (gerade im Heranwachsendenalter) dann letztlich doch zeitlich instabil. Aus diesen sowie aus Datenschutzgründen muss diese letztere Vorgehensweise als recht problematisch angesehen werden.

Steht eine hinreichend verlässliche Validierungsmethode zur Verfügung, kann bei unverschlüsselten Codes ein mehrstufiges, hierarchisches Zuordnungsverfahren angewendet werden, das auf der Anzahl der Fehler bei Beantwortung der Codefragen basiert (Hamming-Ähnlichkeit) und in Pöge (2005b, 2008) beschrieben wird: In einem ersten Schritt werden alle Codepärchen herausgesucht, die eine Übereinstimmung in dem kompletten Code aufweisen (0 Fehler). Alle Pärchen werden nun einer Validierung unterzogen und die sich als passend erwiesenen aus den Daten herausgenommen. Mit den verbleibenden Codes werden weitere Zuordnungs- und Validierungsschritte unter Zulassung von immer mehr Fehlern durchgeführt. Nach unseren Erfahrungen steigt der Kontrollaufwand mit der Anzahl der zugelassenen Fehler deutlich an. Bei einem siebenstelligen Code und der Zulassung von bis zu drei Fehlern, waren bei einer Zuordnung von Duisburger Jugendlichen zwischen den Jahren 2003 und 2004 beispielsweise rund 3.850 Handschriftenkontrollen nötig, um 2.600 echt positive Treffer zu erhalten (Verhältnis rund 1,5 zu 1; Pöge 2008: 65). Neben der Voraussetzung, überhaupt ein Validierunginstrument zur Verfügung zu haben, ist diese Vorgehensweise vor allem aufgrund des hohen Arbeitsaufwandes problematisch. Sowohl die Durchführung der Handschriftenvergleiche als auch die datenverarbeitungstechnische Aufbereitung (Erstellung von Kontrolllisten und Herausnahme der Codes bei validierter Zuordnung bzw. Führen von Listen mit als nicht passend identifizierten Pärchen) ist sehr arbeits- und damit kostenintensiv.

Soll die sich als sehr geeignet herausgestellte Levensthein-Ähnlichkeit (vgl. Schnell/Bachteler/Reiher 2010) angewendet werden oder sollen verschlüsselte Codes zum Einsatz kommen, ist eine hierarchische fehlertolerante Vorgehensweise

11 Dies Verfahren ist nicht unproblematisch, da nach unseren Erkenntnissen viel Erfahrung nötig ist, um zusammengehörige Handschriften zu erkennen. Hier spielt sicherlich das Alter der Befragten und damit zusammenhängend deren veränderliche Handschriften eine Rolle. Aber auch der Einfluss zum Beispiel der Stiftfarbe darf bei den Kontrollen nicht unterschätzt werden.

auf Basis von Fehlern in der Beantwortung der einzelnen Codefragen nicht mehr möglich.[12] Soll dennoch hierarchisch vorgegangen werden, kann man zunächst über die vorgestellten Distanzmaße die Ähnlichkeiten zwischen *allen* Codes zweier Erhebungszeitpunkte bestimmen. Dann kann aus der entstehenden Ähnlichkeitsmatrix pro Code des einen Erhebungszeitpunktes der Code aus dem jeweils anderen Erhebungszeitpunkt ermittelt werden, zu dem die größte Ähnlichkeit besteht.[13] Nun können die Codezuordnungen hierarchisch validiert werden, das heißt, die Codepärchen sollten mit absteigenden Ähnlichkeiten, beispielsweise mit Hilfe von Handschriftenvergleichen der Fragebogen, überprüft werden. Passende Pärchen können herausgeschrieben und die Codes, so sie mit niedrigerer Ähnlichkeit nochmals vorkommen, aus den Daten gelöscht werden. Wird so vorgegangen, stellt sich die Frage, bis zu welchem Ähnlichkeitsniveau der Codepärchen Zuordnungsvalidierungen sinnvollerweise durchgeführt sollten. Im optimalen Fall sollten die Validierungen nur bis zu dem Ähnlichkeitsniveau des unähnlichsten echt positiven Treffers vorgenommen werden. Leider ist dies Niveau in der Praxis unbekannt und man ist auf die Anwendung mehr oder weniger plausibler Schwellenwerte angewiesen. Diese Schwellenwerte entsprechen dann der minimalen Ähnlichkeit, die zwei Codes mindestens aufweisen müssen, um als zusammengehörig identifiziert zu werden.[14]

Der Vorteil dieser Vorgehensweise liegt darin, dass ein beliebiges Ähnlichkeitsmaß verwendet werden kann, welches nicht zwingend eine Entsprechung in der Fehleranzahl der zugrunde liegenden Codes hat, dafür aber möglicherweise besser geeignet ist[15] und/oder auch bei verschlüsselten Codes verwendet werden kann.

Um die Qualität der fehlertoleranten Zuordnung verschlüsselter Codes im Vergleich zu unverschlüsselten zu testen, wird in den nachfolgenden Ausführungen die zuletzt geschilderte Vorgehensweise gewählt. Die Datengrundlage bildet ein selbstgenerierter persönlicher Code, der seit dem Jahr 2002 in der DFG-geförderten Panel-

12 Bei der Levensthein-Ähnlichkeit können Codepärchen mit gleicher Fehlerzahl in den zugrunde liegenden Codefragen unterschiedliche Ähnlichkeiten aufweisen, und bei der Verwendung von verschlüsselten Codes gehen die Informationen über die Fehler in den Codefragen gänzlich verloren. Als Ausnahme gilt hier allerdings die Zuordnung exakt gleicher Codes. Diese Pärchen weisen mit Levenshtein-Ähnlichkeit bzw. in verschlüsselter Form ebenso wie mit der Hamming-Ähnlichkeit immer die maximale Ähnlichkeit auf.

13 Die Praxis zeigt, dass dann allerdings noch eine nicht unerhebliche Anzahl an falsch positiven Treffern identifiziert wird. Existieren nämlich Codes, die zu keinem Code aus dem jeweils anderen Datensatz gehören, werden diese Codes dennoch herausgeschrieben, da sie zu irgendeinem Code eine größtmögliche Ähnlichkeit aufweisen (und sei sie auch Null).

14 Im Falle der Hamming-Ähnlichkeit und unverschlüsselten Codes entspricht dies dem oben vorgestellten Verfahren.

15 Die unten vorgestellten Analysen zeigen beispielsweise, dass die Levenshtein-Ähnlichkeit der Hamming-Ähnlichkeit bei unverschlüsselten Codes deutlich überlegen ist. Schnell, Bachteler und Reiher (2010) schlagen als Schwellenwert für die Mindestähnlichkeit zweier Codes bei der Levenshtein-Distanz einen Wert von 0,34 vor, was einer Levenshtein-Ähnlichkeit von 0,66 entspricht.

studie „Kriminalität in der modernen Stadt“[16] eingesetzt wird (Pöge 2005a, 2007; Pollich 2010) und stetig weiterentwickelt und verbessert wurde (Pöge 2005b, 2008). Für die hier durchgeführten Analysen wurde der Code in seiner letzten, siebenstelligen Version verwendet und in einem Experiment mit Bielefelder Studierenden erhoben. Um die tatsächlich zusammengehörigen Codes identifizieren zu können, wurde zusätzlich die Matrikelnummer der Probanden erfragt. Die Beschreibung dieses Feldexperiments und der resultierenden Daten ist Gegenstand des nächsten Abschnittes.

2 Experiment

Am 21. Oktober 2009 wurde das in Abbildung 1 dargestellte Codeblatt den Teilnehmerinnen und Teilnehmern der Pflichtvorlesung „Einführung in die Methoden der quantitativen empirischen Sozialforschung“ zum Ausfüllen vorgelegt. Der Besuch dieser Veranstaltung ist laut Studienablaufempfehlung für das erste Semester vorgesehen und – neben anderen – für die BA-Studiengänge „Soziologie“, „Sozialwissenschaften“ und „Politikwissenschaft“ obligatorisch. Es gaben 354 Studierende ein ausgefülltes Codeblatt ab, bei 3 Blättern wurde die Angabe der Matrikelnummer verweigert (351 gültige Fälle zu t_1).

Ein halbes Jahr später, am 21. April 2010, wurde das leicht modifizierte Codeblatt[17] den Teilnehmerinnen und Teilnehmern der Pflichtvorlesung „Statistik I“ zur Beantwortung vorgelegt. Der Besuch dieser Veranstaltung wird laut Studienablaufempfehlung für das zweite Semester empfohlen und ist ebenfalls – neben anderen – für die BA-Studiengänge „Soziologie“, „Sozialwissenschaften“ und „Politikwissenschaft“ verpflichtend. Bei einem regulären Studienablauf sollte der Teilnehmerkreis der beiden Veranstaltungen demnach in weiten Teilen deckungsgleich sein. Zum zweiten Zeitpunkt füllten 245 Personen ein Codeblatt aus, bei 2 Blättern wurde die Angabe der Matrikelnummer verweigert (243 gültige Fälle zu t_2).

Eine erste Analyse der erhobenen Daten aus beiden Zeitpunkten zeigt zunächst, dass alle aus der Beantwortung der Fragen gebildeten (Komplett-)Codes in t_1 und in t_2 nur je einmal vorkommen. Sie sind also eindeutig und gut geeignet, die Fälle zu identifizieren bzw. zu differenzieren. Ein Problem mit der Identifikation der Personen eines Zeitpunktes gibt es mit dem siebenstelligen Code bei der Anzahl der Teilnehmerinnen und Teilnehmer dementsprechend nicht. Dies ist aufgrund der relativ geringen Teilnehmerzahl und der Länge auch nicht verwunderlich (vgl. Pöge 2005b: 57ff., 2008: 61ff.).

16 Siehe http://www.krimstadt.de/.

17 Zusätzlich zu den vorhandenen Fragen sollte nun auch das Geschlecht und das Geburtsjahr angegeben werden.

Abbildung 1 Das eingesetzte Codeblatt (Zeitpunkt t_1)

Matrikelnummer: _____

Erstellung des persönlichen Codes

Liebe Teilnehmerin, lieber Teilnehmer,

da wir Ihren Fragebogen dem des letzten Jahres ohne Ihren Namen zuordnen wollen, ist es wichtig, dass Sie sich an Ihren persönlichen Code vom letzten Jahr erinnern. Denn nur so können Ihre Fragebögen einander zugeordnet werden, ohne dass jemand herausfinden kann, wer diese Fragebögen ausgefüllt hat. Wichtig ist also, dass Sie denselben Code noch wissen. Aus diesem Grund haben wir die nachfolgenden Fragen formuliert, die Ihnen helfen sollen, sich an Ihre persönliche Kombination zu erinnern.

Bitte kreuzen Sie bei jeder der sieben Fragen immer nur ein Feld an!
Wenn Sie eine der Fragen überhaupt nicht beantworten können, kreuzen Sie bitte kein Feld an!

Hier nun die sieben Fragen zur Erstellung Ihres persönlichen Codes:

1	Bitte kreuzen Sie den **ersten** Buchstaben des Vornamens Ihres **Vaters** (oder einer Person, die für Sie einem Vater am nächsten kommt) an. (z. B. Anton, Bernd, Hans-Peter usw.). a b c d e f g h i j k l m n o p q r s t u v w x y z ä ö ü ß
2	Bitte kreuzen Sie den **ersten** Buchstaben des Vornamens Ihrer **Mutter** (oder einer Person, die für Sie einer Mutter am nächsten kommt) an. (z. B. Anna, Beate, Jutta, Maria, usw.). a b c d e f g h i j k l m n o p q r s t u v w x y z ä ö ü ß
3	Bitte kreuzen Sie den **ersten** Buchstaben Ihres **Vornamens** an (z. B. Michael, Thomas, Ute usw.). a b c d e f g h i j k l m n o p q r s t u v w x y z ä ö ü ß
4	Bitte kreuzen Sie den **Tag** Ihres **Geburtsdatums** an (z. B. Geburtstag am 7. Januar = 7, am 12. Mai = 12, am 31. Oktober = 31). 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
5	Bitte kreuzen Sie den **letzten** Buchstaben Ihrer natürlichen **Haarfarbe** an. (z. B. braun, Glatze, schwarz, usw.). a b c d e f g h i j k l m n o p q r s t u v w x y z ä ö ü ß
6	Bitte kreuzen Sie den **letzten** Buchstaben Ihrer **Augenfarbe** an. (z. B. braun, grün, grau, usw.). a b c d e f g h i j k l m n o p q r s t u v w x y z ä ö ü ß
7	Bitte kreuzen Sie den **letzten** Buchstaben Ihres **Nachnamens** an (Sollten Sie Ihren Namen gewechselt haben, nehmen Sie Ihren Geburtsnamen!) a b c d e f g h i j k l m n o p q r s t u v w x y z ä ö ü ß

Um zu überprüfen, wie viele Fehler die Befragten im Vergleich der beiden Erhebungszeitpunkte beim Ausfüllen der Codeblätter machten bzw. wie viele Inkonsistenzen in den einzelnen Fragen auftreten, wurden die Fälle aus beiden Zeitpunkten mit gleichen Matrikelnummern einander zugeordnet. Wie Tabelle 1 entnommen werden kann, beläuft sich die Gesamtzahl der Personen auf n=187. Auch wenn

nicht völlig auszuschließen ist, dass hier Fälle zugeordnet wurden, die durch einen Fehler gleiche Matrikelnummern aufweisen, aber nicht zusammengehören, erscheint dies doch äußerst unwahrscheinlich. Im weiteren Verlauf der Analysen werden die genannten 187 Pärchen als tatsächlich zusammengehörend behandelt.

Vergleicht man die Codes, die diese Pärchen zwischen t_1 und t_2 aufweisen, kann zunächst die Anzahl der Beantwortungsfehler analysiert werden. Das Ergebnis dieser Analyse findet sich in Tabelle 1 – es fällt relativ ernüchternd aus. Von den 187 Personen waren lediglich drei Viertel (75,4 %) in der Lage, in beiden Erhebungszeitpunkten die sieben Codefragen gleich zu beantworten. Vergleicht man diese Ergebnisse mit denen aus der oben genannten Schüleruntersuchung, in der ein sehr ähnlicher Code verwendet wurde, ist die Quote jedoch deutlich besser: Bei der Zuordnung zwischen den Jahren 2002/2003 waren nur 45,3 % der 16-jährigen Münsteraner Jugendlichen bei einem ähnlichen (fünfstelligen jedoch schwierigeren) Code zu fehlerfreier Beantwortung in der Lage (vgl. Pöge 2005b: 54). In Duisburg 2003/2004 lag die Quote mit 58,3 % im Vergleich zu Münster deutlich höher, wobei hier durchschnittlich 15-jährige Schülerinnen und Schüler mit einem sechsstelligen Code befragt wurden (vgl. Pöge 2008: 65). Wenn man allerdings berücksichtigt, dass an dem hier durchgeführten Experiment nur Personen mit deutlich höherem Bildungsgrad (Studierende mit zumeist Abitur) und deutlich höherem Alter (durchschnittlich 23 Jahre zu t_2) teilnahmen und sich den bewusst einfachen Charakter der Fragen verdeutlicht, ist das Ergebnis absolut gesehen überraschend schlecht. Es verdeutlicht noch einmal drastisch das generelle Problem der Zuordnung über persönliche Codes und die Notwendigkeit einer fehlertoleranten Zuordnungsmethode.

Tabelle 1 Anzahl der Fehler bei der Beantwortung der Codefragen zu beiden Entstehungszeitpunkten (n=187)

Fehlerzahl	Anzahl	Prozent
0	141	75,4
1	39	20,9
2	5	2,7
3	2	1,1
Gesamt	187	100,0

Eine differenziertere Betrachtung der Fragen, bei denen Beantwortungsfehler gemacht wurden, offenbart, dass deren Anzahl durchaus abhängig vom „Schwierigkeitsgrad" der jeweiligen Fragen ist (siehe Tabelle 2). Die Fragen nach dem *ersten Buchstaben* der Elternvornamen, des eigenen Vornamens und des Tages des Geburtsdatums funktionieren augenscheinlich recht zuverlässig. Die Fehlerquote

liegt hier bei maximal 1,6 %. Demgegenüber waren die Fragen nach den jeweils *letzten Buchstaben* der eigenen Haar- und Augenfarbe aber auch des eigenen Nachnamens deutlich weniger erfolgreich (Fehlerquote zwischen 6,4 und 10,7 %, siehe auch Yurek/Vasey/Havens 2008). Obwohl man annehmen sollte, dass alle Fragen von der hier befragten Klientel korrekt zu beantworten sein müssten, ist indes schlicht zu konstatieren, *dass dies nicht der Fall ist.* Hauptgrund mögen Konzentrations- oder Motivationsprobleme sein, die aller Wahrscheinlichkeit nach jedoch nicht spezifische Probleme des hier geschilderten Experimentes sind. Vielmehr lässt sich vermuten, dass diese Probleme in dem überwiegenden Teil aller realen Befragungssituationen zutage treten. Jedenfalls ist nicht verwunderlich, dass bei den genannten Schülerbefragungen mit deutlich jüngeren Jugendlichen aus allen Schulformen die Ergebnisse in Bezug auf die Reproduktion des Codes noch deutlich schlechter als die der Studierenden sind.[18]

Tabelle 2 Fehler bei der Beantwortung der einzelnen Codefragen (*n*=187)

Frage		Anzahl	Prozent
1	Vorname Vater (erster Buchstabe)	3	1,6
2	Vorname Mutter (erster Buchstabe)	2	1,1
3	eigener Vorname (erster Buchstabe)	1	0,5
4	Tag Geburtsdatum	2	1,1
5	eigene Haarfarbe (letzter Buchstabe)	20	10,7
6	eigene Augenfarbe (letzter Buchstabe)	15	8,0
7	eigener Nachname (letzter Buchstabe)	12	6,4

Es stellt sich mit den vorherigen Resultaten die Frage, welche Auswirkungen bzw. Verzerrungen die oben genannten Sachverhalte mit sich bringen. Um den Befragungsaufwand möglichst gering zu halten, wurde in dem hier durchgeführten Experiment allerdings – neben der Frage nach dem Geburtsjahr – einzig die Frage nach dem Geschlecht der Befragten mit erhoben.[19] Eine entsprechende Auswertung zeigt die schon bekannten Befunde (vgl. Pöge 2005b: 66): Frauen gelingt es deutlich besser, die Codefragen zu zwei Zeitpunkten gleich zu beantworten (siehe Tabelle 3).

18 Am Rande sei erwähnt, dass sich hier die Fragen aufdrängen, wie korrekt inhaltliche Fragen mit meistenteils deutlich höheren Schwierigkeitsgraden generell in schriftlichen Umfragen beantwortet werden und welche Auswirkungen die sicherlich auftretenden Fehler in Bezug auf die Datenqualität haben.

19 Unsere Erkenntnisse aus den Schülerbefragungen lassen eine Verzerrung im Hinblick auf den Bildungsgrad (Schulform) der Befragten vermuten (vgl. Pöge 2005b: 66), die hier aufgrund der diesbezüglichen Homogenität der Stichprobe ohnehin nicht messbar wäre.

Tabelle 3 Fehler bei der Beantwortung der Codefragen nach Geschlecht

	Fehler											
	0		1		2		3		Gesamt		Gesamt t_2	
Geschlecht	*n*	%	*n*	%	*n*	%	*n*	%	*n*	%	*n*	%
weiblich	80	57,1	17	43,6	1	20,0	1	50,0	99	53,2	120	49,6
männlich	60	42,9	22	56,4	4	80,0	1	50,0	87	46,8	122	50,4
Gesamt	140	100,0	39	100,0	5	100,0	2	100,0	186	100,0	242	100,0

Während wir in der Gesamtdatei des Zeitpunktes t_2 ein nahezu paritätisches Geschlechterverhältnis vorfinden, ist dies bei den Personen, die keinen Fehler bei der Beantwortung machten, deutlich in Richtung der Frauen verschoben (Frauen: 57,1 %; Männer: 42,9 %). Nur durch eine fehlertolerante Zuordnungsmethode ist es möglich, das diesbezügliche Missverhältnis zu verbessern. Das Geschlechterverhältnis aller 186 zusammengehörenden Fälle (mit gültigen Angaben bei der Frage nach dem Geschlecht), die nur durch das Zulassen von bis zu 3 Fehlern bei den Codefragen ermittelt werden können, beträgt 53,2 % (Frauen) zu 46,8 % (Männer). Die Differenz zu dem Verhältnis des Gesamtdatensatzes (t_2) lässt sich vermutlich durch ein verzerrtes generelles Verweigerungsverhalten erklären. Anscheinend haben mehr Männer die Teilnahme komplett verweigert oder das Studium abgebrochen, so dass sie an der zweiten Erhebung nicht mehr teilnahmen.

3 Analyse

Um einen Test der Zuordnungsmethoden (unverschlüsselt vs. verschlüsselt) durchzuführen, wird eine Vorgehensweise gewählt, die derjenigen im tatsächlichen Anwendungsfall entspricht oder entsprechen könnte (siehe Abschnitt 1.3). Das bedeutet, die Anwenderin bzw. der Anwender hat vorab keine Kenntnis darüber, wie viele Personen, die zu t_1 befragt wurden, zu t_2 ebenfalls an der Befragung teilnahmen. Die mögliche Vorgehensweise wurde in Abschnitt 1.3 vorgestellt und wird in allen durchgeführten Versuchen gleichermaßen angewandt. Sie besteht aus vier Schritten:

Schritt 1: Berechnung der Ähnlichkeit zwischen allen Fällen aus t_1 und allen Fällen aus t_2 auf Grundlage des gewählten Distanzmaßes.[20]

Schritt 2: Zu jedem Fall aus t_1 wird der Fall aus t_2 ermittelt, zu dem auf Grundlage des jeweiligen Distanzmaßes die größte Ähnlichkeit besteht. Gibt es mehrere Pärchen mit maximaler Ähnlichkeit, werden sie alle verwendet. Die Datei wird abgespeichert und ausgewertet.

Schritt 3: Analog wird zu jedem Fall aus t_2 der Fall aus t_1 bestimmt, zu dem auf Grundlage des jeweiligen Distanzmaßes die größte Ähnlichkeit besteht. Gibt es mehrere Pärchen mit maximaler Ähnlichkeit, werden sie alle verwendet. Die Datei wird abgespeichert und ausgewertet.

Schritt 4: Beide Dateien werden zusammengeführt, Duplikate werden entfernt; danach wird die Gesamtdatei ausgewertet.

Im realen Anwendungsfall würde sich, wie in Abschnitt 1.3 geschildert, eine (hierarchische) Zuordnungsvalidierung der herausgeschriebenen Codepärchen, beispielsweise über Handschriftenvergleiche, anschließen. Auf diesen Schritt wird hier aus Zeit- und Kostengründen verzichtet. Stattdessen soll die Anzahl der ermittelten Codepärchen für einen Vergleich der Leistungsfähigkeit der jeweiligen Zuordnungsmethoden verwendet werden. Je geringer die Anzahl der als zusammengehörig herausgeschriebenen Codes ist, je geringer also die Anzahl der in der Praxis zu validierenden Codepärchen ist, desto effizienter ist das Verfahren, sofern auch alle echt positiven Treffer identifiziert werden.

Um zu verdeutlichen, warum die Schritte 2 *und* 3 drei nötig sind, das heißt die Zuordnungsrichtung von t_1 nach t_2 ($t_1 \rightarrow t_2$) und von t_2 nach t_1 ($t_1 \leftarrow t_2$) bzw. die Zusammenführung der Kombinationen ($t_1 \leftrightarrow t_2$) sei hier ein fiktives Beispiel vorgestellt: Zum Zeitpunkt t_1 liegen die Fälle a und b, zum Zeitpunkt t_2 die Fälle c und d vor. Die Ähnlichkeiten zwischen diesen vier Fällen seien die in Tabelle 4 dargestellten.

20 Auf ein technisches Problem bei der Durchführung sei hingewiesen: Es ist zu berücksichtigen, dass in Schritt 1 zunächst große Datenmengen anfallen, da zu jedem Fall aus t_1 die Ähnlichkeit zu jedem Fall aus t_2 bestimmt werden muss. In dem hier geschilderten Experiment entstehen Dateien mit 351*243 = 85.293 Fällen. Bei zwei Datensätzen mit je 2.000 Fällen ergäbe sich eine Fallzahl von 4 Millionen. Hier stellt sich die Frage, ob das Statistikprogramm im konkreten Anwendungsfall mit einer solch hohen Fallzahl umgehen kann. Die „Merge Toolbox" ermöglicht eine Beschränkung der auszugebenden Fälle und ist auch daher sehr empfehlenswert.

Tabelle 4 Ähnlichkeiten für ein fiktives Beispiel mit 4 Fällen

	c	d
a	1	3
b	2	4

Geht man nach Schritt 1 vor, sucht also zu jedem Fall aus t_1 den Fall mit der höchsten Ähnlichkeit aus t_2, ergeben sich die Kombinationen a-d und b-d. Geht man umgekehrt vor und sucht zu jedem Fall aus t_2 den Fall mit der höchsten Ähnlichkeit aus t_1, ergeben sich die Kombinationen c-b und d-b, wovon letztere schon vorhanden ist. Mit Schritt 4 würden somit 3 Kombinationen verbleiben (a-d, b-d und c-b), die untersucht werden müssten. Im weiteren Verlauf der hier vorgestellten Analysen wird deutlich, dass dieses Vorgehen tatsächlich nötig ist, um immer die komplette Anzahl der tatsächlich zusammengehörigen Pärchen zu ermitteln.

3.1 Zuordnung der unverschlüsselten Codes

Zunächst wird das oben genannte Vorgehen auf Grundlage des nicht verschlüsselten kompletten Codes und der Hamming- sowie der Levenshtein-Ähnlichkeit durchgeführt. Die Bestimmung der Ähnlichkeiten erfolgt mit Hilfe des oben erwähnten Programmes „Merge Toolbox" (MTB). Die Ergebnisse sind in Tabelle 5 dargestellt. Es findet sich in der ersten Spalte das gewählte Ähnlichkeitsmaß, in der zweiten Spalte die Zuordnungsrichtung, so wie im vorangegangenen Abschnitt beschrieben. In der Spalte „Total" wird die Anzahl der Kombinationen bzw. Pärchen ausgewiesen, die sich für die jeweiligen Zuordnungsrichtungen ergeben. In der ersten Zeile bedeutet der ausgewiesene Wert, dass sich für die 351 Fälle aus t_1 786 Kombinationen ergeben, wenn zu jedem einzelnen Fall aus t_1 die Fälle mit maximaler Ähnlichkeit aus t_2 zugeordnet werden. Es sind deshalb mehr als 351 Fälle, da bei mehrfachem Vorkommen der maximalen Ähnlichkeit alle Kombinationen in den Daten belassen werden.

In der Spalte „Treffer" wird aufgelistet, wie viele korrekte Pärchen bzw. echt positive Treffer in diesen Kombinationen enthalten sind. In der Spalte „Treffer in %" wird der prozentuale Absolutwert dargestellt (187 korrekte Treffer sind es insgesamt, daher entspricht 187 100 %). In der Spalte „Trefferquote in %" findet sich das prozentuale Verhältnis der echt positiven Treffer zu den insgesamt ermittelten Kombinationen („Total"). In der Spalte „kein Treffer" steht die Anzahl der falsch positiven Treffer, was der Differenz zwischen „Total" und „Treffer" entspricht.

Tabelle 5 Ergebnisse für den unverschlüsselten kompletten Code (Ähnlichkeitsmaße: Hamming, Levenshtein)

	Richtung	kein Treffer	Treffer	Treffer in %	Total	Trefferquote in %	minimale Ähnlichkeit Treffer ($ä_{min}$)	Fälle mit $ä \geq ä_{min}$
Hamming	$t_1 \rightarrow t_2$	599	187	100,00	786	23,79	0,571	513
	$t_1 \leftarrow t_2$	252	187	100,00	439	42,60	0,571	330
	$t_1 \leftrightarrow t_2$	789	187	100,00	976	19,16	0,571	600
Levenshtein	$t_1 \rightarrow t_2$	449	187	100,00	636	29,40	0,571	358
	$t_1 \leftarrow t_2$	173	187	100,00	360	51,94	0,571	275
	$t_1 \leftrightarrow t_2$	578	187	100,00	765	24,44	0,571	413

Grau hinterlegt: Versuche, in denen alle Treffer identifiziert werden.

Die Spalte „minimale Ähnlichkeit Treffer ($ä_{min}$)" weist die Ähnlichkeit des unähnlichsten echt positiven Treffers (minimale Ähnlichkeit) aus. Dieser Wert lässt sich für die Hamming-Distanz anschaulich interpretieren: 0,571 ergibt sich nach der oben dargestellten Formel bei 3 Fehlern im siebenstelligen Code (1-3/7) und wie in Tabelle 1 aufgezeigt, beträgt die Maximalzahl der Fehler bei den korrekt zusammengehörigen Pärchen 3.

Für die Zuordnung im praktischen Anwendungsfall bzw. die Zuordnungsvalidierung ist eine weitere Kennzahl bedeutsam: Wie viele Pärchen werden identifiziert, deren Codes eine gleiche bzw. größere Ähnlichkeit zeigen als der unähnlichste echt positive Treffer? Geht man nämlich bei der Zuordnungsvalidierung, wie oben beschrieben, hierarchisch vor, wäre dies die Anzahl der Codezuordnungen, die kontrolliert werden müssten, um alle echt positiven Treffer zu identifizieren. In der Spalte „Fälle mit $ä \geq ä_{min}$" wird daher die Anzahl der Pärchen angegeben, die gleiche oder größere Ähnlichkeit als der unähnlichste echt positive Treffer haben. Beispielsweise existieren 600 Codekombinationen mit einem Ähnlichkeitsniveau von größer/gleich 0,571 (also mit drei oder weniger Fehlern), wenn man die Hamming-Ähnlichkeit und die Zuordnungsrichtung $t_1 \leftrightarrow t_2$ betrachtet. Diese müssten in der Praxis, wie in Abschnitt 1.3 dargestellt, hierarchisch validiert werden. Es gilt dabei prinzipiell: Je niedriger diese Zahl ist, desto besser werden die Treffer von den Nicht-Treffern separiert und desto weniger Kontrollen müssten im praktischen Anwendungsfall durchgeführt werden. Bei der effizienteren Levenshtein-Ähnlichkeit sind es zum Beipiel nicht mehr 600 sondern nur 413 Codekombinationen.

Diese Anzahl reduziert sich in der Praxis allerdings noch dadurch, dass bei der erfolgreichen Validierung eines Codepärchens auf höherem Ähnlichkeitsniveau alle weiteren Zuordnungsvorschläge mit den beiden entsprechenden Codes auf niedrigerem Ähnlichkeitsniveau eliminiert werden können. Der Umfang dieser

Reduzierung ist schwer abzuschätzen. Unserer Erfahrung nach ist es möglich, bei unverschlüsselten Codes und Benutzung der Hamming-Distanz auf einen Wert von 1,5 zu 1 (Handschriftenkontrollen : Treffer) zu kommen (siehe oben). Diese Reduzierung bei hierarchischer Vorgehensweise ergibt sich indes bei allen Distanzmaßen und auch bei der Verwendung von verschlüsselten Codes. Um den zu erwartenden Aufwand für Validierungen zu vergleichen, sollen die in der Spalte „Fälle mit ä ≥ $ä_{min}$" ausgewiesenen Werte als Indikatoren betrachtet und verglichen werden, obwohl sich deren absolute Werte im realen Anwendungsfall noch deutlich reduzieren würden.

Neben der Möglichkeit zur vergleichenden Analyse sind die ausgewiesenen Werte auch im Hinblick auf die Entwicklung von Schwellenwerten bedeutsam, bis zu welchem Ähnlichkeitsniveau eine kosten- und zeitintensive Validierung sinnvoll ist. Das Ziel ist ja, möglichst alle echt positiven Treffer zu identifizieren. Unsere Ergebnisse legen hier mit 0,571 eine untere Grenze für einen Schwellenwert nahe, der die Identifikation falsch negativer Treffer verhindert. Das heißt, es wird mit diesem Wert im vorgestellten Fall ausgeschlossen, dass korrekte Treffer nicht gefunden werden. Dieser Schwellenwert kann in der Praxis allerdings zu vielen falsch positiven Treffern führen. Hier muss im Anwendungsfall entschieden werden, ob der durchzuführende Validierungsaufwand im Verhältnis zu dem zu erwartenden Ertrag an echt positiven Treffern steht und der Schwellenwert unter Umständen nachjustiert werden. Schnell, Bachteler und Reiher (2010) empfehlen für die Levenshtein-Ähnlichkeit einen etwas höheren Wert (0,66), der sich im Hinblick auf einen möglichst guten Ausgleich zwischen den Anzahlen an falsch negativen und falsch positiven Pärchen als günstig erwiesen hat. Dieser Wert schließt jedoch nicht aus, dass korrekte Treffer nicht identifiziert werden.

Vergleicht man die Ergebnisse der Zuordnungen über die Hamming- und die Levenshtein-Ähnlichkeit, kann man festhalten, dass in beiden Fällen alle 187 passenden Pärchen ermittelt werden. Dies gilt für die Zuordnungsrichtungen von t_1 nach t_2, von t_2 nach t_1 und logischerweise auch für die zusammengesetzte Datei aus beiden Zuordnungsrichtungen. Die Zuordnungen über die beiden Ähnlichkeitsmaße unterscheiden sich allerdings nicht unerheblich in der Anzahl der falsch positiven Pärchen (Spalten „kein Treffer") und damit auch in der Gesamtzahl der Pärchen (Spalten „Total" und „Trefferquote in %"). Hier schneidet die Levenshtein-Ähnlichkeit deutlich besser ab. Die Trefferquote, also der prozentuale Anteil der Treffer an der Gesamtzahl, liegt mit 19 % (Hamming) und 24 % (Levenshtein) auf einem recht niedrigen Niveau. Nach den hier dargestellten Ergebnissen könnte es empfehlenswert sein, in einem ersten Schritt nur die Zuordnungsrichtung von t_2 nach t_1 zu betrachten (Zeilen $t_1 \leftarrow t_2$). Es werden alle echt positiven Pärchen identifiziert und die Trefferquote liegt mit 43 % (Hamming) und 54 % (Levenshtein) deutlich höher.

Zu beachten ist allerdings, dass diese Quoten nur im Hinblick auf die theoretisch bestmöglichen Zuordnungsquoten beurteilt werden dürfen. Mit der gewählten Vorgehensweise ist die Minimalzahl der herausgesuchten Pärchen mit maximaler Ähnlichkeit und damit die bestmögliche Quote nämlich vorgegeben (Spalte „Total"): Da zu t_1 351 Fälle vorliegen, stellt diese Zahl das Minimum für $t_1 \rightarrow t_2$ dar (wenn alle Fälle aus t_1 jeweils nur ein Pendant aus t_2 zugewiesen bekommen). Für diese Zuordnungsrichtung kann also auch keine bessere Trefferquote als 53,3 % erreicht werden. Für die Richtung $t_2 \rightarrow t_1$ ist das Minimum 243, da in t_2 243 Fälle enthalten sind. Daher kann hier eine Trefferquote von 77,0 % nicht überschritten werden. Für $t_1 \leftrightarrow t_2$ hängt das Minimum von einer weiteren Bedingung ab. Sind alle 243 Pärchen ($t_1 \leftarrow t_2$) in den 351 ($t_1 \rightarrow t_2$) enthalten, liegt es bei 351. Sind alle 243 nicht in den 351 enthalten, liegt es bei 594, ansonsten im Bereich dazwischen. Das entspricht maximalen Trefferquoten von 53,3 % bis 31,5 %. Die in der Tabelle dargestellten Trefferquoten müssen also immer in Relation zu diesen theoretisch überhaupt nur möglichen Quoten interpretiert werden.[21]

Auch die Indikatoren für die zu erwartende Anzahl der im konkreten Anwendungsfall durchzuführenden Validierungen („Fälle mit ä ≥ $ä_{min}$") zeigen für alle Zuordnungsrichtungen eine deutliche Überlegenheit der Levenshtein-Ähnlichkeit.

Insgesamt stellt sich in den hier dargestellten Ergebnissen eine deutlich bessere Performanz der Levenshtein- gegenüber der Hamming-Ähnlichkeit bei der Zuordnung von unverschlüsselten Codes dar. Bei der Zuordnung von unverschlüsselten Codes sollte daher der Levenshtein-Ähnlichkeit Vorzug vor der Hamming-Ähnlichkeit gegeben werden.

3.2 Zuordnung der verschlüsselten Codes

Um die Leistungsfähigkeit des Zuordnungsverfahrens mit verschlüsselten Codes zu überprüfen und sie mit derjenigen eines Verfahrens über unverschlüsselte Codes zu vergleichen, wurden die in Tabelle 6 aufgelisteten 28 Versuche durchgeführt (84 Teilversuche). Das Programm „BloomEncoder" ermöglicht, wie oben bereits ausgeführt, unter anderem die Variation der Länge der Bloomfilter (Bitsize) und der Anzahl der Hashfunktionen, die zur Verschlüsselung verwendet werden. Daneben kann eingestellt werden, welche Länge die Teilstrings haben sollen, in die der komplette Code aufgesplittet wird („Ngramme") und ob dem ersten und letzten N-Gramm ein Leerzeichen voran- bzw. hintangestellt wird („Padded"). Bei allen hier

21 Das Maximum liegt immer bei 243 * 351 = 85.293, was eine Untergrenze der Trefferquote von 0,002 % ergibt.

durchgeführten Versuchen wird die Länge der Bloomfilter konstant bei 1.000 Bit belassen (Voreinstellung) und die Option „Padded" gesetzt, das heißt, es werden immer Leerzeichen am Anfang und Ende verwendet. Systematisch variiert wird die Anzahl der Hashfunktionen, die beginnend mit 1 stetig erhöht wird. Für jede Hashfunktionsanzahl wird je ein Versuch mit Monogrammen und Bigrammen durchgeführt (Ngramme = 1 bzw. Ngramme = 2). Bei zwei Versuchen (Versuch t3 und v3, Tabelle 6) werden exemplarisch Trigramme verwendet (Ngramme = 3). Ausgewertet werden dann jeweils die Zuordnungsrichtungen $t_1 \rightarrow t_2$, $t_1 \leftarrow t_2$ und $t_1 \leftrightarrow t_2$.

Tabelle 6 Durchgeführte Versuche mit verschlüsselten Codes

Versuch	Bitsize	Hash-Fuktionen	Ngramme	Padded
a	1000	1	1	1
b	1000	1	2	1
c	1000	2	1	1
d	1000	2	2	1
e	1000	5	1	1
f	1000	5	2	1
g	1000	10	1	1
h	1000	10	2	1
i	1000	15	1	1
j	1000	15	2	1
k	1000	20	1	1
l	1000	20	2	1
m	1000	25	1	1
n	1000	25	2	1
o	1000	50	1	1
p	1000	50	2	1
q	1000	100	1	1
r	1000	100	2	1
s	1000	200	1	1
t	1000	200	2	1
t3	1000	200	3	1
u	1000	300	1	1
v	1000	300	2	1
v3	1000	300	3	1
w	1000	400	1	1
x	1000	400	2	1
y	1000	500	1	1
z	1000	500	2	1

Im Ergebnis zeigt sich zunächst, dass bis zu einer Hashfunktionsanzahl von einschließlich 200 in den hier durchgeführten Versuchen (a bis t3) in zumindest einer Variante (N-Gramme und Zuordnungsrichtung) alle zusammengehörigen Fälle auch identifiziert werden (siehe Tabellen 7 und 8, grau hinterlegte Zeilen). Auffällig ist, dass dies jedoch nur bei der Verschlüsselung über Bigramme (Ngramme = 2) und nicht über Monogramme (Ngramme = 1) gelingt. Auch der mit 200 Hashfunktionen durchgeführte Versuch t3 mit Trigrammen (Ngramme = 3) identifiziert nicht alle Treffer korrekt. Mit einer Hashfunktionsanzahl größer als 200 (Versuche u bis z) funktioniert die Methode dann jedoch nicht mehr zuverlässig: In keinem der Versuche mit mehr als 200 Hashfunktionen werden alle zusammengehörigen Pärchen gefunden.

Tabelle 7 Ergebnisse der Versuche a bis p

	Richtung	kein Treffer	Treffer	Treffer in %	Total	Trefferquote in %	Minimale Ähnlichkeit Treffer ($ä_{min}$)	Fälle mit $ä \geq ä_{min}$
a	$t_1 \rightarrow t_2$	291	186	99,47	477	38,99	0,476	–
	$t_1 \leftarrow t_2$	93	185	98,93	278	66,55	0,476	–
	$t_1 \leftrightarrow t_2$	358	186	99,47	544	34,19	0,476	–
b	$t_1 \rightarrow t_2$	282	186	99,47	468	39,74	0,444	–
	$t_1 \leftarrow t_2$	84	187	100,00	271	69,00	0,444	246
	$t_1 \leftrightarrow t_2$	339	187	100,00	526	35,55	0,444	352
c	$t_1 \rightarrow t_2$	242	186	99,47	428	43,46	0,741	–
	$t_1 \leftarrow t_2$	80	186	99,47	266	69,92	0,741	–
	$t_1 \leftrightarrow t_2$	300	186	99,47	486	38,27	0,741	–
d	$t_1 \rightarrow t_2$	215	186	99,47	401	46,38	0,500	–
	$t_1 \leftarrow t_2$	70	187	100,00	257	72,76	0,500	221
	$t_1 \leftrightarrow t_2$	261	187	100,00	448	41,74	0,500	265
e	$t_1 \rightarrow t_2$	188	186	99,47	374	49,73	0,727	–
	$t_1 \leftarrow t_2$	67	184	98,40	251	73,31	0,727	–
	$t_1 \leftrightarrow t_2$	236	186	99,47	422	44,08	0,727	–
f	$t_1 \rightarrow t_2$	176	186	99,47	362	51,38	0,487	–
	$t_1 \leftarrow t_2$	57	187	100,00	244	76,64	0,487	220
	$t_1 \leftrightarrow t_2$	214	187	100,00	401	46,63	0,487	267
g	$t_1 \rightarrow t_2$	170	185	98,93	355	52,11	0,738	–
	$t_1 \leftarrow t_2$	62	183	97,86	245	74,69	0,727	–
	$t_1 \leftrightarrow t_2$	219	186	99,47	405	45,93	0,727	–
h	$t_1 \rightarrow t_2$	170	185	98,93	355	52,11	0,509	–
	$t_1 \leftarrow t_2$	59	186	99,47	245	75,92	0,509	–
	$t_1 \leftrightarrow t_2$	213	187	100,00	400	46,75	0,509	270
i	$t_1 \rightarrow t_2$	171	185	98,93	356	51,97	0,742	–
	$t_1 \leftarrow t_2$	62	182	97,33	244	74,59	0,729	–
	$t_1 \leftrightarrow t_2$	218	186	99,47	404	46,04	0,729	–
j	$t_1 \rightarrow t_2$	168	185	98,93	353	52,41	0,540	–
	$t_1 \leftarrow t_2$	56	187	100,00	243	76,95	0,540	216
	$t_1 \leftrightarrow t_2$	209	187	100,00	396	47,22	0,540	254
k	$t_1 \rightarrow t_2$	167	186	99,47	353	52,69	0,743	–
	$t_1 \leftarrow t_2$	62	183	97,86	245	74,69	0,743	–
	$t_1 \leftrightarrow t_2$	214	186	99,47	400	46,50	0,743	–
l	$t_1 \rightarrow t_2$	165	186	99,47	351	52,99	0,552	–
	$t_1 \leftarrow t_2$	56	187	100,00	243	76,95	0,552	216
	$t_1 \leftrightarrow t_2$	206	187	100,00	393	47,58	0,552	250
m	$t_1 \rightarrow t_2$	167	185	98,93	352	52,56	0,749	–
	$t_1 \leftarrow t_2$	62	183	97,86	245	74,69	0,739	–
	$t_1 \leftrightarrow t_2$	213	186	99,47	399	46,62	0,739	–
n	$t_1 \rightarrow t_2$	166	185	98,93	351	52,71	0,564	–
	$t_1 \leftarrow t_2$	56	187	100,00	243	76,95	0,564	216
	$t_1 \leftrightarrow t_2$	206	187	100,00	393	47,58	0,564	256
o	$t_1 \rightarrow t_2$	166	185	98,93	351	52,71	0,785	–
	$t_1 \leftarrow t_2$	62	181	96,79	243	74,49	0,785	–
	$t_1 \leftrightarrow t_2$	212	185	98,93	397	46,60	0,785	–
p	$t_1 \rightarrow t_2$	165	186	99,47	351	52,99	0,642	–
	$t_1 \leftarrow t_2$	56	187	100,00	243	76,95	0,642	214
	$t_1 \leftrightarrow t_2$	205	187	100,00	392	47,70	0,642	245

–: Nicht berechnet, da nicht alle Treffer identifiziert werden.
Grau hinterlegt: Versuche, in denen alle Treffer identifiziert werden.

Tabelle 8 Ergebnisse der Versuche q bis z

	Richtung	kein Treffer	Treffer	Treffer in %	Total	Treffer-quote in %	Minimale Ähnlichkeit Treffer ($ä_{min}$)	Fälle mit $ä \geq ä_{min}$
q	$t_1 \rightarrow t_2$	169	182	97,33	351	51,85	0,841	–
	$t_1 \leftarrow t_2$	60	183	97,86	243	75,31	0,826	–
	$t_1 \leftrightarrow t_2$	213	184	98,40	397	46,35	0,826	–
r	$t_1 \rightarrow t_2$	165	186	99,47	351	52,99	0,744	–
	$t_1 \leftarrow t_2$	56	187	100,00	243	76,95	0,744	215
	$t_1 \leftrightarrow t_2$	207	187	100,00	394	47,46	0,744	264
s	$t_1 \rightarrow t_2$	172	182	97,33	354	51,41	0,879	–
	$t_1 \leftarrow t_2$	65	178	95,19	243	73,25	0,894	–
	$t_1 \leftrightarrow t_2$	214	182	97,33	396	45,96	0,879	–
t	$t_1 \rightarrow t_2$	165	186	99,47	351	52,99	0,866	–
	$t_1 \leftarrow t_2$	60	183	97,86	243	75,31	0,866	–
	$t_1 \leftrightarrow t_2$	217	187	100,00	404	46,29	0,866	360
t3	$t_1 \rightarrow t_2$	172	179	95,72	351	51,00	0,861	–
	$t_1 \leftarrow t_2$	69	174	93,05	243	71,60	0,875	–
	$t_1 \leftrightarrow t_2$	233	182	97,33	415	43,86	0,861	–
u	$t_1 \rightarrow t_2$	176	177	94,65	353	50,14	0,895	–
	$t_1 \leftarrow t_2$	69	174	93,05	243	71,60	0,937	–
	$t_1 \leftrightarrow t_2$	224	177	94,65	401	44,14	0,895	–
v	$t_1 \rightarrow t_2$	178	173	92,51	351	49,29	0,912	–
	$t_1 \leftarrow t_2$	147	163	87,17	310	52,58	0,933	–
	$t_1 \leftrightarrow t_2$	322	178	95,19	500	35,60	0,912	–
v3	$t_1 \rightarrow t_2$	203	148	79,14	351	42,17	0,967	–
	$t_1 \leftarrow t_2$	91	153	81,82	244	62,70	0,944	–
	$t_1 \leftrightarrow t_2$	293	156	83,42	449	34,74	0,944	–
w	$t_1 \rightarrow t_2$	184	173	92,51	357	48,46	0,950	–
	$t_1 \leftarrow t_2$	73	172	91,98	245	70,20	0,946	–
	$t_1 \leftrightarrow t_2$	232	174	93,05	406	42,86	0,946	–
x	$t_1 \rightarrow t_2$	330	165	88,24	495	33,33	0,950	–
	$t_1 \leftarrow t_2$	199	156	83,42	355	43,94	0,955	–
	$t_1 \leftrightarrow t_2$	525	172	91,98	697	24,68	0,950	–
y	$t_1 \rightarrow t_2$	13380	172	91,98	13552	1,27	0,961	–
	$t_1 \leftarrow t_2$	11839	173	92,51	12012	1,44	0,959	–
	$t_1 \leftrightarrow t_2$	13671	175	93,58	13846	1,26	0,959	–
z	$t_1 \rightarrow t_2$	19443	164	87,70	19607	0,84	0,973	–
	$t_1 \leftarrow t_2$	16713	163	87,17	16876	0,97	0,965	–
	$t_1 \leftrightarrow t_2$	25045	172	91,98	25217	0,68	0,965	–

–: Nicht berechnet, da nicht alle Treffer identifiziert werden.
Grau hinterlegt: Versuche, in denen alle Treffer identifiziert werden.

In Bezug auf die Zuordnungsrichtung ist festzustellen, dass bei der Richtung $t_1 \rightarrow t_2$ in keinem Versuch alle 187 tatsächlich zusammengehörigen Fälle ermittelt werden. In der Gegenrichtung $t_1 \leftarrow t_2$ werden in deutlich mehr Fällen die korrekten Treffer identifiziert. Bei den Versuchen mit Bigrammen im Bereich von 1 bis 200

Hashfunktionen gelingt dies in 8 von 10 Versuchen (b, d, f, j, l, n, p und r). In allen 10 Versuchen mit Bigrammen im Bereich von 1 bis 200 Hashfunktionen werden alle 187 korrekten Treffer nur in der zusammengesetzten Datei (Zuordnungsrichtung $t_1 \leftrightarrow t_2$) identifiziert (b, d, f, h, j, l, n, p, r und t).

Betrachtet man die Gesamtzahl der ermittelten Pärchen mit maximaler Ähnlichkeit (Spalte „Total"), dann zeigt sich schon im diesbezüglich schlechtesten Versuch mit allen korrekt ermittelten Treffern (Versuch b) eine deutliche Verbesserung zur Zuordnung mit unverschlüsselten Codes (siehe Tabelle 5). So treten bei der Zuordnungsrichtung $t_1 \leftrightarrow t_2$ im verschlüsselten Fall (Bigramme, 1 Hashfunktion) 526 Pärchen auf, in denen die 187 korrekten Treffer enthalten sind (Quote: 35,55 %). Im unverschlüsselten Fall waren dies bei der Hamming-Ähnlichkeit 976 Pärchen (Quote: 19,16 %) und bei der Levenshtein-Ähnlichkeit 765 Pärchen (Quote: 24,44 %). Dieses Verhältnis wird bei zunehmender Zahl von Hashfunktionen (im Bereich bis 200 Funktionen) immer besser zu Gunsten der Zuordnung über verschlüsselte Codes. Das beste Ergebnis wird hier bei Versuch p erreicht (Bigramme, Bitsize = 1.000, 50 Hashfunktionen): Bei Zuordnungsrichtung $t_1 \leftrightarrow t_2$ werden lediglich 392 Pärchen ermittelt, in denen die 187 korrekten Treffer enthalten sind (Quote: 47,70 %). Offensichtlich wirkt die Verschlüsselung hier positiv differenzierend, ohne dass die Fähigkeit, die korrekten Treffer zu identifizieren, verloren geht. Damit zusammenhängend ist die Anzahl der Pärchen mit gleicher oder größerer Ähnlichkeit als der Treffer mit der minimalsten Ähnlichkeit im besagten Versuch p am niedrigsten.[22] Hier liegen 245 Pärchen vor, die die 187 korrekten Treffer enthalten. Im konkreten Anwendungsfall müssten also maximal nur 245 Pärchen mit einem Ausschuss von 58 falsch positiven Treffern kontrolliert werden, um alle korrekten Treffer zu erhalten.[23] Bei der Zuordnung über unverschlüsselte Codes liegt die Anzahl der maximal zu kontrollierenden Pärchen bei 600 (Hamming) bzw. 413 (Levensthein) und damit in beiden Fällen erheblich höher.

In den hier analysierten Versuchen mit einer Hashfunktionsanzahl größer 200 funktioniert die Zuordnung nicht mehr befriedigend, denn es werden nicht mehr alle korrekten Treffer identifiziert. Dies liegt höchstwahrscheinlich an dem Verhältnis von Hashfunktionsanzahl und Länge der Bloomfilter.[24] Es werden zu viele Einsen in die Filter geschrieben, so dass die Ähnlichkeit in allen Fällen zu sehr an-

22 Diese Anzahl wurde nur bestimmt, wenn auch alle Treffer korrekt ermittelt wurden.

23 Auch hier gilt, dass sich diese Anzahl bei hierarchischer Vorgehensweise noch weiter reduzieren kann.

24 Zwar wird bei der Ermittlung des Dice-Koeffizienten die Länge der Filter nicht explizit berücksichtigt, diese gibt jedoch die Anzahl der möglichen Stellen vor, an denen überhaupt Einsen gespeichert werden können. Es kommt daher bei zu kurzen Filtern in Bezug auf die Hashfunktionszahl vermutlich häufig zu Kollisionen.

steigt, um eine genügende Differenzierung zwischen fehlerhaften und korrekten Codepärchen zu ermöglichen. Ein deutliches Indiz dafür ist, dass ab 300 Hashfunktionen die Versuche mit Monogrammen eine höhere Trefferquote haben als die mit Bigrammen. Bei der Aufsplittung der Codes in Bigramme und nachfolgender Verschlüsselung über die Hashfunktionen werden nämlich mehr Einsen in die Filter geschrieben als bei Monogrammen.

4 Fazit und Ausblick

Die hier vorgestellten Ergebnisse zeigen das zunächst erstaunliche Resultat, dass die Zuordnung von verschlüsselten Codes der Zuordnung von unverschlüsselten deutlich überlegen ist. Werden Bigramme und eine nicht zu hohe Anzahl an Hashfunktionen bei der Verschlüsselung in Bloomfilter der Länge 1.000 Bit verwendet, so ist die Rate der falsch positiven Treffer deutlich kleiner, ohne dass die Treffergenauigkeit leidet. Auch im Hinblick auf die im Anwendungsfall durchzuführenden Validierungen (beispielsweise mit Handschriftenvergleichen) ist eine deutlich niedrigere Anzahl zu erwarten.

In den durchgeführten Versuchen zeigte sich die beste Performanz bei einer Hashfunktionsanzahl von 50 mit dem verwendeten siebenstelligen[25] Code. Diese Anzahl reicht im Zusammenspiel mit den anderen Parametern (Bigramme, Bitsize = 1.000) völlig aus, um eine zureichende Verschlüsselungssicherheit zu realisieren. Die Empfehlung von 15 Funktionen (Schnell/Bachteler/Reiher 2009a: 211) kann auf Basis der hier vorgestellten Resultate deutlich nach oben korrigiert werden.

Falls im konkreten Anwendungsfall die Anzahl der falsch positiven Treffer weiter gesenkt werden muss, beispielsweise um einen eventuellen Kontrollaufwand über Handschriftenvergleiche etc. zu minimieren, kann auf Grundlage der vorliegenden Erkenntnisse eine Einschränkung der Zuordnungsrichtung in Betracht gezogen werden. Hierbei ist es sinnvoll, die kleinere von zwei vorliegenden Dateien als Grundlage zu verwenden. Hier war dies die Datei aus dem zweiten Erhebungszeitpunkt t_2. Allerdings muss einschränkend hinzugefügt werden, dass in den durchgeführten Versuchen bei einer solchen Einschränkung nicht immer alle korrekten Treffer identifiziert wurden. Will man diesbezüglich sichergehen, müssen beide Zuordnungsrichtungen beachtet werden.

25 Zu beachten ist, dass bei der Aufsplittung in N-Gramme der eigentlich siebenstellige Code in manchen Fällen zu einem achtstelligen wird. Dies ist der Fall, wenn der Tag des Geburtstages zweistellig ist. In den Analysen zeigte sich, dass dies Verhalten einen positiven und erwünschten Effekt hat, daher wurde auf eine komplette Umstellung auf sieben Stellen, durch Voranstellen einer Null bei einstelligen Geburtstagen, verzichtet.

Als untere Schranke des Ähnlichkeitsniveaus, bis zu dem eine Validierung sinnvollerweise durchgeführt werden sollte, legen unsere Ergebnisse bei der unverschlüsselten Zuordnung (Hamming- und Levenshtein-Ähnlichkeit) einen Schwellenwert von 0,571 nahe. Bei der verschlüsselten Zuordnung ist ein solcher Schwellenwert schwierig anzugeben, da er augenscheinlich von den gewählten Verschlüsselungsparametern abhängt. Zumindest weist in allen durchgeführten Versuchen kein korrekter Treffer eine Ähnlichkeit von unter 0,444 auf. Insofern ist dieser Wert möglicherweise eine realistische untere Schranke. Die angegebenen Schwellenwerte stehen dabei unter der Prämisse, dass auf keinen echt positiven Treffer verzichtet werden soll. Kann dies hingegen akzeptiert werden, bietet sich an, die Werte nach oben zu verschieben, um den Validierungsaufwand zu verringern.

Alles in allem scheint der Einsatz des vorgestellten Verfahrens bei der Zuordnung des hier verwendeten verschlüsselten Codes uneingeschränkt empfehlenswert. Die Ergebnisse fordern geradezu, solch einer Zuordnung den Vorzug gegenüber einer Methode mit unverschlüsselten Codes zu geben. Inwieweit die Befunde auf den Einsatz und die Zuordnung anderer selbstgenerierter Codes übertragbar sind, ist mit den vorliegenden Analysen allerdings nicht zu beantworten. Es kann vermutet werden, dass die Ergebnisse auf ähnliche, selbstgenerierte persönliche Codes übertragbar sind. Um dies zu überprüfen, sind weitere Untersuchungen hilfreich und sinnvoll. Die von Schnell, Bachteler und Reiher unentgeltlich zur Verfügung gestellten Programme „Merge Toolbox", „BloomEncoder" und „BloomComparator" ermöglichen in jedem Fall die komfortable und zuverlässige Durchführung eines solchen Vorhabens.

Literatur

Dice, L. R., 1945: Measures of the Amount of Ecologic Association Between Species. Ecology 26 (3): 297-302.

Galanti M. R., R. Siliquini, L. Cuomo, J. C. Melero, M. Panella, F. Faggiano, and the EU-DAP study group, 2007: Testing Anonymous Link Procedures for Follow-Up of Adolescents In A School-Based Trial: The EU-DAP pilot study. Preventive Medicine 44 (2): 174-177.

Grube, Joel W., M. Morgan und K. A. Kearney 1989: Using Self-Generated Identification Codes to Match Questionnaires in Panel Studies of Adolescent Substant Use. Addictive Behaviors 14 (2): 159-171.

Hamming, R. W., 1950: Error Detecting and Error Correcting Codes. The Bell System Technical Journal 26 (2): 147-160.

Leitgöb, H., 2010: Klassifikation von Verläufen mittels Optimal Matching. S. 475-492 in: J. Bacher, A. Pöge und K. Wenzig (Hg.): Clusteranalyse. Anwendungsorientierte Einführung in Klassifikationsverfahren. 3. Aufl. München: Oldenbourg.

Levenshtein, V. I., 1966: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Cybernetics and Control Theory 10 (8): 707-710.

Pöge, A., 2005a: Methodendokumentation der kriminologischen Schülerbefragung in Münster 2000-2003 (Vier-Wellen-Panel). Bd. 9. Schriftenreihe „Jugendkriminalität in der modernen Stadt – Methoden". Münster, Trier.

Pöge, A., 2005b: Persönliche Codes bei Längsschnittstudien: Ein Erfahrungsbericht. ZA-Information 56: 50-69. http://www.gesis.org/fileadmin/upload/forschung/publikationen/zeitschriften/za_information/ZA-Info-56.pdf (15.4.2011).

Pöge, A., 2007: Methodendokumentation der kriminologischen Schülerbefragung in Duisburg 2002 bis 2005 (Vier-Wellen-Panel). Bd. 13. Schriftenreihe „Jugendkriminalität in der modernen Stadt – Methoden". Münster, Bielefeld.

Pöge, A., 2008: Persönliche Codes ›reloaded‹. Methoden – Daten – Analysen. Zeitschrift für Empirische Sozialforschung 2 (1): 59-70. http://www.gesis.org/fileadmin/upload/forschung/publikationen/zeitschriften/mda/Vol.2_Heft_1/2008_MDA1_Poege.pdf (15.4.2011).

Pollich, Daniela, 2010: Methodendokumentation der kriminologischen Schülerbefragung in Duisburg 2002 bis 2007 (Sechs-Wellen-Panel). Bd. 16. Schriftenreihe „Jugendkriminalität in der modernen Stadt – Methoden". Münster, Bielefeld.

Schnell, R., T. Bachteler und J. Reiher, 2005: MTB: Ein Record-Linkage-Programm für die empirische Sozialforschung. ZA-Information 56: 93-103. http://www.gesis.org/fileadmin/upload/forschung/publikationen/zeitschriften/za_information/ZA-Info-56.pdf (15.4.2011).

Schnell, Rainer, T. Bachteler und J. Reiher, 2009a: Entwicklung einer neuen fehlertoleranten Methode bei der Verknüpfung von personenbezogenen Datenbanken unter Gewährleistung des Datenschutzes. Methoden – Daten – Analysen. Zeitschrift für Empirische Sozialforschung 3 (2): 203-217. http://www.gesis.org/fileadmin/upload/forschung/publikationen/zeitschriften/mda/Vol.3_Heft_2/06_Schnell_et_al.pdf (15.4.2011).

Schnell, R., T. Bachteler und J. Reiher, 2009b: Privacy-Preserving Record Linkage Using Bloomfilters. BMC Medical Informatics and Decision Making 41 (9).

Schnell, R., T. Bachteler und J. Reiher, 2010: Improving the Use of Self-Generated Identification Codes. Evaluation Review 34 (5): 391-418.

Yurek, L. A., J. Vasey und D. S. Havens, 2008: The Use of Self-Generated Identification Codes in Longitudinal Research. Evaluation Review 32 (5): 435-452.

Anschrift des Autors

Akad. Rat Dr. Andreas Pöge
Universität Bielefeld
Fakultät für Soziologie
Postfach 10 01 31
33501 Bielefeld
andreas.poege@uni-bielefeld.de